Optimización de curvas ROC para perceptrones multicapa mediante técnicas evolutivas

Juan Carlos Pazos Mandiá

Trabajo Fin de Estudios para
el grado de Ingeniería Informática

en

Ciencia de la Computación e Inteligencia Artificial
Universidad Carlos III Madrid

Tutores:
Prof. Ricardo Aler Mur
Prof. Inés María Galván

Junio 2015

Índice general

Índice de tablas

Capítulo 1

Introducción

1.1. Problemas de clasificación

Los procesos de clasificación basados en aprendizaje automático juegan un papel importante en diversas áreas de aplicación. Por ejemplo, la meteorología, con clasificadores de nubes identificándolas para conocer posibles niveles de precipitación. La industria, con clasificación de materiales obteniendo su resistencia. La música, con clasificadores de canciones, obteniendo el estilo musical basándose en la información sonora. La publicidad digital, clasificando cada consumidor potencial y realizando una campaña dirigida. La medicina, clasificando pacientes sanos o enfermos en función de analíticas, etc.

Los problemas de clasificación requieren de instancias o ejemplos, que serán agrupadas en función de los atributos o características que manifiesten en clases previamente fijadas. Cuando una clasificación permite únicamente la agrupación en 2 posibles clases, se denomina clasificación binaria. En cambio, si se trata de una donde existe la posibilidad de diferenciar en más de 2 clases, se trata de clasificación multiclase.

En algunos entornos, la proporción de instancias que pertenezcan a determinada clase puede ser significativamente superior respecto al resto de las clases. En estos casos el problema de clasificación se encuentra desbalanceado. Este tipo de problemas se da con frecuencia en los dominios médicos.

Los problemas de clasificación más complejos son aquellos en los que los patrones o ejemplos de diferentes clases se entremezclan, requiriendo de clasificadores no lineales. Son en estas situaciones donde las técnicas de aprendizaje automático adquieren importancia en cuanto a su posibilidad de realizar una clasificación no lineales de los patrones.

1.2. Técnicas de aprendizaje automático

El aprendizaje automático engloba múltiples técnicas pertenecientes al ámbito de la inteligencia artificial, donde se utilizan ejemplos o instancias para obtener de manera automática un modelo que represente dicho conjunto de instancias. Este tipo de aprendizaje es capaz de realizarse de manera automática partiendo de dichas experiencias, que han de ser obtenidas de manera empírica.

Las técnicas de esta disciplina permiten extraer características y patrones a partir de la información disponible, capaces de generar un modelo con el que representar el funcionamiento buscado.

Existen dos principales tipos de aprendizaje automático, el supervisado y el no supervisado.

- **Supervisado:** Los algoritmos de aprendizaje automático supervisado reciben el conjunto de datos a clasificar y la clase a la que pertenece cada una de las instancias a predecir. De ahí la definición de supervisado, ya que el aprendizaje está controlado por los datos de las clases. Gracias a esta información, el algoritmo será capaz de inferir una función por la cual predecir, ya sea de manera directa o continua (función de regresión), la clase para cualquier instancia indicada como entrada.

  Las redes de neuronas artificiales son un ejemplo de técnica de aprendizaje automático de tipo supervisado, muy capaces de resolver problemas de clasificación, además de problemas de regresión, debido a su versatilidad en la modelización de una función representativa de la distribución no lineal de las diferentes clases.

- **No Supervisado:** Las técnicas de aprendizaje automático no supervisado, intentan agrupar y diferenciar los sectores en los que se pueden dividir el conjunto de datos que se le indique como entrada. Estos algoritmos, al carecer de valores sobre la pertenencia de las instancias a determinadas clases, son capaces de encontrar las relaciones entre las distintas experiencias, consiguiendo modelar un sistema que las organice en agrupaciones.

Este proyecto se encuadra en el tipo de aprendizaje supervisado.

1.3. Curva ROC

La manera más común de evaluar un clasificador es mediante la tasa de aciertos (o de error). Pero en ocasiones, se está interesado en desglosar su rendimiento para cada una de las clases que forman el dominio. Esto es especialmente cierto en problemas donde hay muchos mas datos de una clase que de la otra y en las que se puede obtener una buena tasa de error global a costa de fallar en la clase minoritaria. Una herramienta bastante establecida para problemas de dos clases, que no tiene los problemas mencionados, son las curvas ROC, en las que se muestra la tasa de aciertos de una de las clases frente a la tasa de errores de la otra. Cuanto más pronunciada sea la curva, indicará un mayor acierto en la clasificación de ambas clases.

La forma habitual de valorar la pronunciación de la curva ROC es mediante el cálculo del área que existe bajo ella, de este modo se puede apreciar en forma de valor numérico cuan óptimo es el método de clasificación y realizar una correcta valoración de manera más sencilla. El área bajo la curva ROC tomará valores desde 0 a 1 e indica la probabilidad de que la clase otorgada a cada instancia sea la correcta; un área bajo la curva superior al 0.5 (50 %) indica que tiene una capacidad discriminatoria superior al azar.

1.4. ROC y RNA

Como anteriormente se ha comentado, las redes neuronales pueden ser utilizadas para clasificación. Mediante el algoritmo de aprendizaje propio de esta técnica, el descenso del gradiente, las instancias se procesan conociendo la clase a la que pertenecen. El objetivo del aprendizaje es minimizar el error cuadrático obtenido al clasificar cada una de las instancias del conjunto de datos disponibles.

La salida de la red neuronal es un valor continuo, por lo que cuando se utilizan para resolver problemas de clasificación es necesario definir un umbral que determinará la pertenencia de las instancias a una clase o a otra.
Para las redes de neuronas, al igual que para cualquier clasificador, es posible construir la curva ROC asociada a la red.
Para ello es necesario representar gráficamente el porcentaje de instancias correctamente clasificadas para cada uno de los puntos de corte que se establezcan como umbral posibles. Es decir si el clasificador da como salida una distribución continua entre 0 y 1, los umbrales con los que se calculará la curva ROC, tendrán el mismo recorrido.
Para pintarla se clasificará como clase 0 a todas las salidas inferiores al valor de umbral, y como clase 1 a las salidas superiores al mismo, de esta forma la curva continua que se representa es denominada curva ROC. Es decir, para cada valor del umbral hay un punto en la curva ROC. Moviendo el umbral de 0 a 1 se generan todos los puntos de la curva ROC.

El entrenamiento típico de una red consigue minimizar el error cuadrático medio de la misma, pero esto no implica necesariamente la optimización de la curva ROC. Con ella se puede evaluar la capacidad de clasificación de una red de neuronas ya entrenada mediante las correctas e incorrectas clasificaciones del modelo. Puede suceder que una red entrenada carezca de una curva ROC apropiada. Este problema puede ser más notorio en conjuntos de datos desbalanceados en los que no existe la misma proporción de instancias para cada una de las clases del conjunto. En este contexto los algoritmos evolutivos puede ser una alternativa a los métodos tradicionales porque pueden permitir optimizar el área bajo la curva ROC directamente.

1.5. Algoritmos Evolutivos

En problemas como éste es donde toman importancia los algoritmos evolutivos, ayudando a las técnicas existentes de clasificación a mejorar su rendimiento. En concreto gracias a su generalidad se puede utilizar una técnica evolutiva como optimizador de una red para que modificando los pesos de la

misma, que tendrán su representación en el evolutivo como un parámetro denominado cromosoma, se maximice el área bajo la curva ROC. Esto lo consiguen alterando sus pesos mediante pequeñas mutaciones con el objetivo de optimizar la función de fitness.

Una de sus características más preciada para este trabajo, se trata de la capacidad de modificar la función a optimizar, por lo que en éste caso será la función encargada de maximizar el área bajo la curva ROC. El evolutivo realizará evaluaciones mediante dicha función para decidir qué pasos realizar en las siguientes iteraciones.

1.6. Objetivo del proyecto

El objetivo de este trabajo será realizar un estudio para determinar si las curvas ROC que se obtienen a partir de una red de neuronas pueden ser mejoradas mediante el uso de computación evolutiva. Para ello se codificarán en el cromosoma los pesos y umbrales de la red. Para lo que se utilizará como función de fitness a optimizar el área bajo la curva ROC asociada al clasificador obtenido de la red.

En este proyecto se realizarán las validaciones utilizando diversos dominios, todos ellos con dos clases diferentes a las que pueden pertenecer las correspondientes instancias de cada uno de los mismos. Todos ellos están desbalanceados en mayor o menor medida, con el objetivo de validar el sistema de optimización desarrollado en estas situaciones.

Para conseguir el objetivo del proyecto se ha desarrollado un sistema, para el cual se han realizado las siguientes tareas:

Como tareas específicas para la elaboración completa de este proyecto se pueden distinguir:

Estudio previo
: Comprensión del funcionamiento global del método de optimización a diseñar, así como de cada una de las partes que lo compondrán.

Preprocesado de dominios
: Los diferentes conjuntos de datos pertenecientes a cada uno de los dominios serán tratados para que las clases de cada una de sus instancias estén representadas por 1 ó 2. Para realizar un correcto entrenamiento de la red neuronal es necesario que las instancias con las que se realizará el aprendizaje estén desordenadas aleatoriamente. Además, se empleará validación cruzada por lo que se tendrán que generar diferentes grupos, también denominados folds, del mismo dominio y para evitar producir un aprendizaje erróneo, todos ellos deben tener un porcentaje similar de instancias para cada clase.

Entrenamiento de una red de neuronas
: Entrenar una red neuronal cuya configuración esté compuesta por un número de entradas (idéntico al número de variables del problema), una neurona de salida y un número de neuronas ocultas, fijada a priori.

Cálculo de curva ROC

Calcular la curva ROC de una red de neuronas, variando el umbral de clasificación

Aplicación de un algoritmo evolutivo

Se ha decidido utilizar un algoritmo ya existente, CMAES, por lo que ha sido necesario realizar una correcta integración del algoritmo en el entorno de trabajo. Para ello se deberá descomponer la configuración de la red en forma de vector que será utilizado como cromosoma del evolutivo.

Función fitness basada en la curva ROC

Para la aplicación del algoritmo evolutivo utilizado, CMAES, es necesario definir la función de fitness que cuantificará lo óptimo que es el individuo. Dado que CMAES considera mejor un individuo cuya evaluación se aproxime a cero, ésta deberá cumplir con dicho requisito. $1 - AUC$

Unificación del sistema

Realizar una integración de las diferentes partes previamente mencionadas.

Generar proceso de experimentación

Se ha realizado una experimentación en diferentes dominios, variando el número de neuronas ocultas y la semilla aleatoria inicial. Se ha automatizado la ejecución de todos los experimentos. Además se han generado procesos para almacenar los resultados de la experimentación.

Comparación de los resultados

Se ha realizado un análisis de los resultados, comparando la eficiencia del sistema cuando la red de neuronas utilizada por el algoritmo evolutivo es inicializada aleatoriamente versus a inicializar el algoritmo con una red ya entrenada.

1.7. Estructura de la memoria

Redes de neuronas artificiales:

En este capítulo se tratarán las redes de neuronas de manera mucho más detallada, centrándose en explicar el perceptrón multicapa, que será el tipo de red neuronal que se ha utilizado para desarrollar el proyecto. Se mostrará su arquitectura y el mecanismo de aprendizaje.

Estrategias evolutivas:

En este capítulo se procederá a explicar el funcionamiento de los algoritmos evolutivos, centrándonos fundamentalmente en el la estrategia evolutiva CMAES, que será el utilizado para optimizar los pesos de la red neuronal para este proyecto.

Curvas ROC:

En este capítulo se presentan todos los conceptos relativos a curvas ROC. Se incluye también el mecanismo para calcular la curva ROC para el caso a estudiar (perceptrón multicapa).

Sistema de optimización:

En este capítulo se presentará el sistema de optimización desarrollado, los componentes separados del mismo, su lógica y funcionamiento, así como sus distintos estados. Además, se indicarán las herramientas de software utilizadas para conseguir dicho fin.

Validación experimental:

En este capítulo se presentan los diferentes dominios utilizados para la validación experimental. Se detallarán los resultados obtenidos del proceso de optimización de los clasificadores con diferentes semillas, y se mostrará información relevante así como un análisis de los mismos.

Conclusiones y futuros trabajos:

En este capítulo se incluyen las conclusiones obtenidas del proyecto, así como problemas encontrados y cómo se abarcaron durante el periodo de trabajo. También se mostrarán las futuras lineas por las que se puede plantear una continuación del trabajo obtenido.

Capítulo 2

Redes de Neuronas

Las redes de neuronas artificiales son un modelo de aprendizaje que tiene como base el sistema nervioso de un ser vivo, formado por las neuronas que reciben información y tienen la capacidad de producir un estímulo de salida, todas las neuronas del modelo están pensadas para colaborar entre ellas gracias a estar interconectadas.
Cada una de estas neuronas son las unidades básicas de la red, las cuales constan de una serie de entradas y una salida, por donde propagarán su estímulo, éste viene dado por diversas funciones:

- Función de propagación: Por lo general es el sumatorio de los valores de entrada a dicha neurona multiplicados por su valor de interconexión con la misma.
- Función de activación: Que en caso de no existir se aplicaría sólo la de propagación.
- Función de transferencia: Esta función se aplica sobre el valor obtenido de la función de propagación, y su objetivo es el de limitar el rango de los posibles valores a tomar. Las dos funciones más usadas para este cometido son la tangente hiperbólica $tanh(x) = \frac{sinh(x)}{cosh(x)}$, que que acota la salida en el rango [-1, 1], y la función sigmoidal $sigmoide(x) = \frac{1}{1+e^{-t}}$, que acota la salida en el rango [0, 1].

En función del tipo de organización y del tipo de aprendizaje existen diferentes tipos de redes neuronales:

- Topología:
  - Monocapa: Como el perceptrón simple.
  - Multicapa: Como el perceptrón multicapa, Que será el tipo de red que se use para este proyecto.
- Aprendizaje:
  - Supervisado: Necesita un conjunto de datos cuyas salidas de la red ya son conocidas para poder realizar el aprendizaje.

- No supervisado: No necesitan ningún conjunto previo, Son capaces de autoorganizar los datos que recibe. Como las redes de Kohonen
- Redes híbridas: Muy útiles como aproximadores universales. Como las redes de base radial.
- Por refuerzo: Se realiza este tipo de aprendizaje al poseer una función que denote si la decisión tomada por la red ha sido buena o mala, Como Q-Learning.

Gracias a la capacidad de algunos tipos de redes neuronales de realizar el aprendizaje a partir de un conjunto de ejemplos, se puede lograr generar una aproximación a los resultados deseados. Para los problemas de clasificación y regresión se comportan de forma excelente los tipos de redes con capacidad de generar aproximaciones no lineales, por lo que abordarán de manera exitosa dichos problemas. Como por ejemplos los perceptrones multicapa o las redes de base radial.

2.1. Perceptrón Multicapa

El perceptrón multicapa es la siguiente evolución del perceptrón simple, el cual es capaz de representar únicamente problemas lineas. La incorporación de capas intermedias en la red nace como requisito para poder afrontar problemas de clasificación no lineal. La capa intermedia de la red otorga esta preciada cualidad.

La arquitectura del perceptrón consta de un número de neuronas en la primera capa (capa de entrada), por donde se introducirán las correspondientes variables de entrada del conjunto de datos representativos del problema. Cada una de las neuronas de entrada está conectada por lo general a todas las neuronas existentes en la capa intermedia de la red. Esta conexión posee un peso por el que será multiplicado el valor procedente de la neurona de entrada, hasta llegar a la neurona de la capa intermedia correspondiente, donde se le sumará el resto de valores que lleguen de otras entradas a la misma neurona intermedia además del umbral de la misma, otro peso más de la red. Además de la función de activación no lineal y de tenerla, la función de transferencia asociada a dicha neurona. Finalmente, todas las neuronas de la capa intermedia están conectadas a las diferentes neuronas de salida. Para estas neuronas se sigue el mismo procedimiento de cálculo que en las dos primeras capas, multiplicando los valores y sumándolos en la neurona de salida, además de su correspondiente umbral la función de activación y la de transferencia.
En la siguiente figura se muestra un ejemplo de la arquitectura que representa un perceptrón multicapa con 3 neuronas de entrada, 5 neuronas en la capa oculta y 1 de salida.

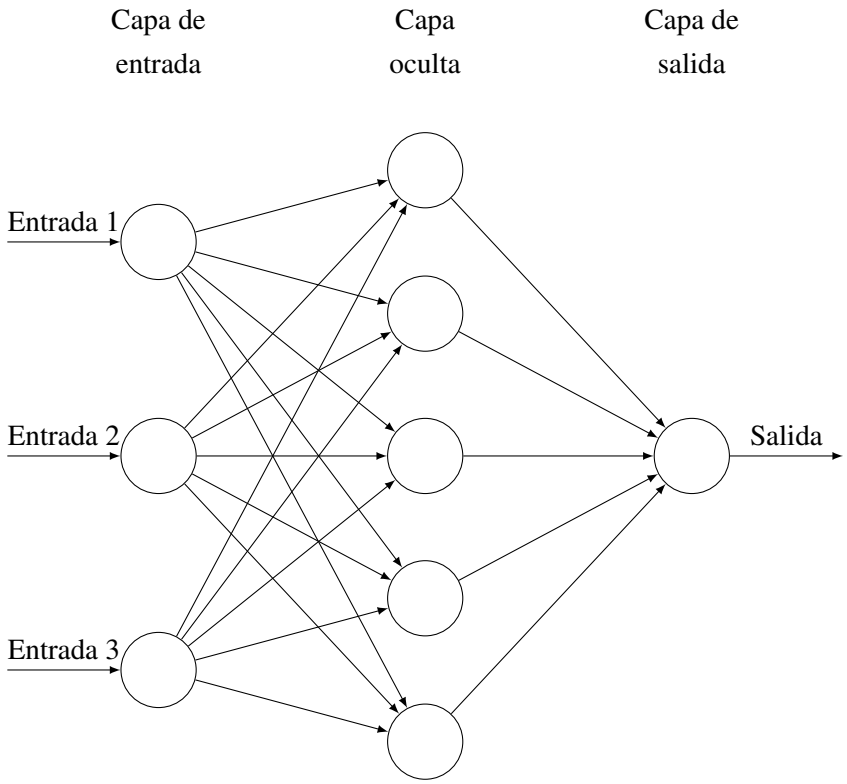
Como se explicó en la sección anterior, las funciones de activación que se suelen utilizar en el Perceptron Multicapa son la tangente hiperbólica o la función sigmoidal. Con la primera se logra obtener unos resultados comprendidos entre -1 y 1 y mediante el uso de la segunda, el valor de la salida se situará entre 0 y 1.

El aprendizaje del perceptrón multicapa se lleva a cabo para minimizar el error cuadrático medido en la salida de la red. Este error conocido como ECM se calculará haciendo la media para todas las instancias del cuadrado de la diferencia entre el valor de salida obtenido y el valor deseado. Siendo n el

número de instancias, Y el vector de las salidas obtenidas y S el vector de las salidas deseadas la fórmula para calcular dicho error quedaría así.

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(Y_i - S_i)^2$$

Para realizar el aprendizaje se aplica la regla delta generalizada o algoritmo de retropropagación, que básicamente consiste en ir modificando los pesos de las neuronas en base al ECM y retropropagándolo hacia las neuronas ocultas y las neuronas de entrada.



ESTRUCTURA PERCEPTRON MULTICAPA

Capítulo 3

Estrategias Evolutivas

Introducción a los algoritmos evolutivos cómo funcionan, principales aplicaciones y tipos.

Las técnicas evolutivas se basan en la representación computacional de la teoría de la evolución. En general, estos algoritmos almacenan una población de individuos, donde cada uno de ellos representa una posible solución al problema a resolver. Cada individuo sufrirá transformaciones ya sea junto a otros individuos de la población o mutaciones individuales. También deberán pasar por procesos de selección, tras los cuales sólo continuarán para la siguiente generación los mejores.
Tras realizar múltiples generaciones o iteraciones del algoritmo los individuos de la población restante, estarán muy cerca de la solución óptima al problema.

Los algoritmos evolutivos poseen como punto fuerte la capacidad de combinar la búsqueda dirigida de la solución gracias a los procesos de selección en la población, y la búsqueda aleatoria gracias a los diferentes procesos de transformación que sufren los individuos.

En ellas se emplea como individuos una codificación de variables que puedan resolver el problema, en un vector de números reales. Como algoritmos de transformación de individuos utilizan cruce, mutación y operación de selección que puede transformar la población por medios deterministas o probabilísticos, abandonando los individuos que estén por debajo de la capacidad de resolución del problema promedio que tenga la población. Este tipo de selección es determinista, eligiendo los mejores individuos de entre los descendientes $((\mu/\rho, \lambda) - ES)$ o de entre los mejores del conjunto unión de padres y descendientes $(\mu/\rho + \lambda) - ES)$ siendo μ el número de padres, ρ el número de individuos elegidos para evolucionar y λ la descendencia.

3.1. CMAES

El nombre de este algoritmo: CMAES, son las siglas de (Covariance Matrix Adaptation Evolution Strategy). Es un algoritmo de estrategia evolutiva para problemas de difícil optimización en dominios de

distribución continua. Está caracterizado por el uso de la matriz de covarianza como guía en la búsqueda del individuo óptimo.

Para ello CMAES estima una distribución probabilística partiendo de los individuos con mejores resultados con la intención de orientar la búsqueda hacia las regiones del espacio de clasificación que puedan tener resultados más prometedores. Esta distribución probabilística se calcula mediante la normal multivariante $N(m, \sigma^2 C)$ donde m es la media que representa la situación actual en la búsqueda y se moverá hacia mejores situaciones conforme mejoren los individuos; y $\sigma^2 C$ es la matriz de covarianza que controla la mutación y es usada para orientar la búsqueda. En cierto modo la matriz de covarianza apunta hacia mejores soluciones basándose en los individuos anteriores que obtuvieron buenos resultados mediantes la función de fitness asociada para la búsqueda, es decir, los que obtuvieron un valor más bajo con dicha función.

Los pasos a seguir por el algoritmo de CMAES son los siguientes:
1. Inicializar los parámetros m, σ, and C
2. Por cada nueva generación. (i=0,1,2,3...)
(a) Se calculan λ individuos muestreando la distribución $N(m, \sigma^2 C) : x_i \leftarrow N(m, \sigma^2 C)$
(b) Se evalúa el fitness para cada uno de los individuos.
(c) Se actualizan los parámetros de distribución m, σ y C en base a los mejores resultados $x_1, ..., x_\mu$

Capítulo 4

Curvas ROC

La calidad de un diagnóstico para clasificación de instancias entre dos clases se basa fundamentalmente en la capacidad que tenga dicho diagnóstico en distinguir entre ambas, tratando de minimizar al máximo la incertidumbre del diagnóstico dado.
La evaluación de la calidad de dicha asignación se realiza mediante la exactitud del método, que representa su capacidad para diferenciar ambos estados.
Por lo general los resultados otorgados por un clasificador siguen una distribución en escala continua, por lo que presenta la necesidad de definir un valor de corte en la misma, usado para realizar la separación entra ambas clases.

La única manera existente de valorar la exactitud del método a estudiar es mediante experimentación, analizando los conjuntos de datos del dominio que se quiera evaluar. Mediante esta evaluación los posibles resultados a obtener son asignaciones correctas a la primera clase, asignaciones incorrectas a la primera clase, asignaciones correctas a la segunda clase y asignaciones incorrectas a la segunda clase. Para representarlo de una forma más sencilla se muestra la llamada $Tabla\ de\ contingencia$ también denominada $Matriz\ de\ confusión$

		CLASE REAL 1	CLASE REAL 2
Clase Predicha	1	CORRECTA CLASE 1	INCORRECTA CLASE 1
	2	INCORRECTA CLASE 2	CORRECTA CLASE 2

DATOS 4.1: TABLA DE CONTINGENCIA

Mediante el uso de esta tabla se pueden realizar las evaluaciones necesarias para generar la curva ROC, y que para su representación únicamente serán necesarios los valores de la tasa de correctos clasificados para cierta clase y la de falsos clasificados para la misma.

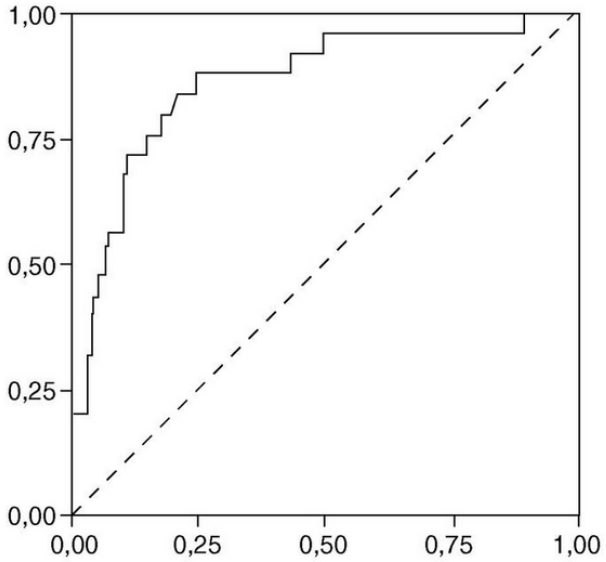


Foto 4.1:

Cuanto mejor sea la clasificación entre los distintos estados, más exacto será el método de asignación y más cerca del vértice superior izquierdo estará la curva ROC.

Las curvas ROC por lo general están comprendidas desde la esquina inferior izquierda, hasta la superior derecha. La línea diagonal imaginaria que las une indica el mínimo que la curva de dicho método debe sobrepasar, ya que si se queda por debajo indicaría una incorrecta asignación de clases y se podría solucionar en la mayoría de los casos modificando el criterio de asignación de clase, de mayor que, a menor que o viceversa.

Este tipo de curvas son fácilmente comparables, tan sencillo como decir que la que más se acerque arriba a la izquierda es más exacta, y dan información clara sobre la especificidad y la sensibilidad.

Una de las maneras más sencillas de representar gráficamente una curva ROC, y la que se usará para la red en este trabajo, es de manera empírica; obteniendo la matriz de confusión para todos los puntos de corte que se puedan considerar para el conjunto a evaluar. Cada nueva evaluación con el nuevo punto de corte se dibujará una linea, que en caso de aumentar el número de correctos clasificados en la clase 1, será vertical, de lo contrario (incorrectamente clasificado en la clase 1) la linea tomará una dirección horizontal. Un tipo de curva como ésta, que se basa en datos en vez de en parámetros es calificada como una curva ROC no paramétrica.
De ser un clasificador discreto (devuelve positivo o negativo) se representa como un punto en el espacio ROC.

Como implementación para conseguir realizar una medición global del método de diagnóstico a evaluar, se recurre al calcular el área bajo la curva ROC, cuanto mayor sea este área, mas arriba a la izquierda está la representación de la curva (cosa que se puede ver gráficamente)

El área bajo la curva ROC, será el indicador al que se recurrirá durante la investigación. Será capaz de indicar entre dos posibles métodos de diagnóstico, cual de ellos otorga la mayor exactitud a la prueba dado el conjunto actual de pacientes a evaluar.

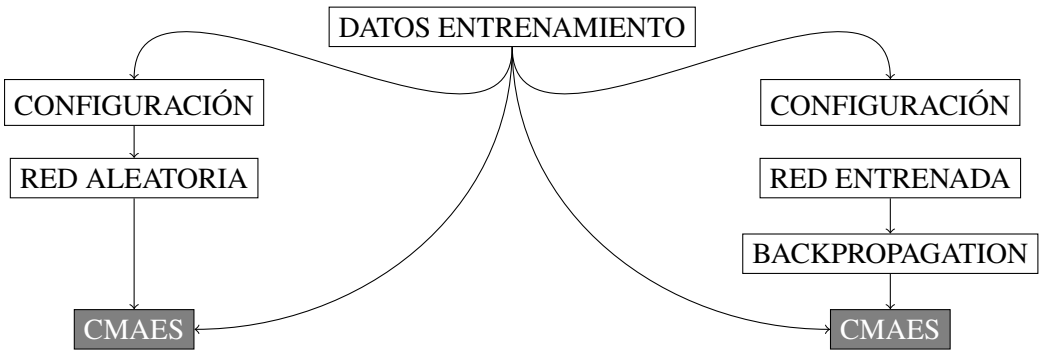
Capítulo 5

Sistema de optimización

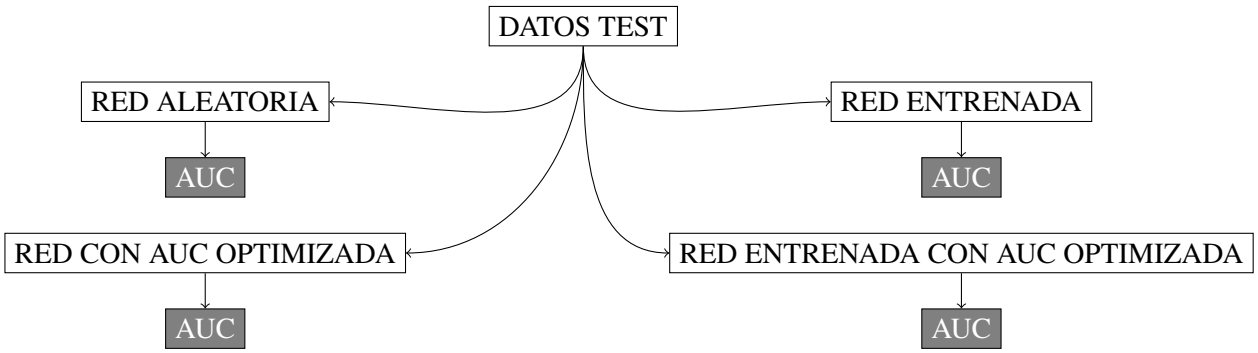
A continuación se explicará como se han aunado todos los elementos para conseguir el sistema capaz de cumplir el objetivo principal, para ello se irá obteniendo paso a paso cada uno de los objetivos específicos comentados con anterioridad.

5.0.1. Arquitectura del sistema

El sistema consta de dos fases principales: entrenamiento y test. La primera se encarga de la estructuración de la red, que podrá ser entrenada con un conjunto de datos o meramente inicializada y posteriormente se realizará la optimización con el evolutivo.
La segunda se encarga de la evaluación de los resultados obtenidos, ya que para cada experimentación se obtendrán resultados para la red neuronal inicializada aleatoriamente, para la optimización mediante el evolutivo de dicha red, para la red entrenada y para la optimización de la red entrenada.



FASE DE ENTRENAMIENTO



FASE DE ENTRENAMIENTO

5.1. Diseño del sistema

Este sistema se encargará de dado un conjunto de datos relativo a un dominio en concreto, realizar un aprendizaje logrando maximizar el área bajo la curva ROC. Por lo que es importante para su correcta elaboración que dichos conjuntos de datos puedan ser correctamente interpretables.

Ya que se recurrirá a usar la validación cruzada para cada ejecución del sistema de optimización, los datos deberán estar divididos en tantos grupos como folds se especifiquen por parámetros. Para que la validación cruzada se pueda realizar de forma correcta, es necesario que cada uno de los grupos contenga una proporción representativa de cada una de las diversas clases que existan en el dominio. Esto quiere decir que cada clase deberá presentarse en aproximadamente la misma proporción entre los distintos grupos. Para llevar a cabo este requisito se recurre a la distribución de las instancias aleatoriamente entre grupos, teniendo cada clase una semilla distinta asignada. Teniendo en cuenta que la distribución aleatoria sea uniforme, estadísticamente, cada fold deberá contener la misma cantidad de experiencias de cada clase que el resto.

Una vez creados los distintos folds con los que se realizará la validación cruzada, se aleatorizan las instancias comprendidas en los mismos, para favorecer en un futuro la capacidad de generalización de la red neuronal con la que se realice el aprendizaje.

A continuación los datos deberán pasar por la red neuronal. El conjunto de datos a emplear depende de la iteración de validación cruzada en la que se encuentre el proceso. En caso de que fuera una validación cruzada de 5 folds, y se encontrara en la segunda ejecución, se utilizarían para conjunto de entrenamiento los folds 1,3,4 y 5 y para test el fold 2.

Una vez identificada la correspondencia de cada fold, se agrupan todos los pertenecientes a entrenamiento en uno y de igual manera sucede con los destinados a hacer test. Tras este proceso se realiza un sesgo, separando los valores de la instancia y la clase esperada para los mismos. Los valores de la instancia pasarán a llamarse valores de entrada a la red, y el valor identificador de la clase correspondiente se identificará como salida esperada de la red.

Para mayor comodidad con el entrenamiento de la red neuronal se decide normalizar los valores de salida entre 0 y 1.

Con los conjuntos de datos listos para recorrer la red neuronal, se procede a configurarla. Esta configuración vendrá dada por la parametrización del sistema, pudiendo seleccionar únicamente el número de neuronas en la capa oculta (intermedia), ya que las neuronas de entrada y las de salida están condicionadas por los atributos de las instancias de cada dominio.

Hay que tener en cuenta que la salida de la red neuronal deberá estar comprendida entre 0 y 1, como se ha indicado anteriormente. Para ello se indicará que la función de salida de la red será la tangente sigmoidal, por la que se conseguirá obtener un rango [0,1] en los valores de salida obtenidos por la red, coincidiendo con el rango de salidas deseado.

Entre los valores que se podrán configurar del multicapa se encuentran la tasa de aprendizaje, el número de ciclos, la tasa de entrenamiento (que indicará el porcentaje del conjunto de datos introducidos para entrenamiento que se utilizará con tal fin) y la tasa de test (que indicará el porcentaje del conjunto de datos introducidos para entrenamiento que se utilizará con el fin de realizar test sobre el sistema). Estos dos últimos valores se usarán para controlar el criterio de para de la red neuronal, cuando el error obtenido del subconjunto dedicado al test de la red, alcance unos valores de error aceptables como para finalizar el entrenamiento.

Llegado a este punto, el sistema sería capaz de dado un conjunto de datos realizar n iteraciones de entrenamiento para una red neuronal con diferentes conjuntos. Siendo n el número de folds configurados para la validación cruzada. Para poder almacenar la información de la red entrenada, se toma lo que sería la identidad de la red, un vector que contiene los valores de todos los pesos de la red entrenada, por expresarlo de algún modo, posee la identidad de la red que ha sido entrenada.

A continuación, ya que en este proyecto se busca conocer si se puede llegar a realizar una optimización de red tomando como objetivo maximizar la curva ROC. Será necesario tener una forma de evaluar cuán pronunciada es la curva ROC para dicha red. Para saberlo se deberá calcular la curva ROC basándose en los diferentes conjuntos destinados para entrenamiento y para test de cada iteración de la validación cruzada y ejecutar una simulación de la red usando los pesos anteriormente nombrados para configurarla y reobtener el sistema conseguido en el entrenamiento. Una vez obtenida la curva, se procederá a calcular el área bajo la misma.

Como paso final, se deberá implementar un algoritmo evolutivo para trabajar con los pesos de la red neuronal ya entrenada. Al que se le deberá indicar el conjunto de pesos desde el que partir el aprendizaje. Dado que para este proyecto se utilizará CMAES como algoritmo evolutivo, hay que tener en cuenta que el objetivo de CMAES aplicado a este caso concreto es minimizar el valor de fitness obtenido mediante una función que se le indique por parámetro. Ya que la meta del proyecto es maximizar el área bajo la curva y la de CMAES es minimizar la función de fitness, si se le indica como función de fitness a CMAES (1 - Área bajo la curva ROC), el minimizar dicha función implicaría maximizar el área bajo la curva. De

esta manera se estaría persiguiendo el objetivo deseado.

De la misma manera que anteriormente para la red neuronal, se realizarán simulaciones tanto para el conjunto de entrenamiento como para el de test indicados por los folds de la validación cruzada.

Para tener valores representativos de estos experimentos se recurrirá a realizar la misma evaluación de cada dominio, con al menos 3 semillas distintas. Para evitar tomar como cierta una posible experiencia anómala.

5.1.1. Función de fitness

El objetivo del algoritmo evolutivo es maximizar el área bajo la curva ROC, y se encarga de minimizar el valor retornado por la función de fitness. Ya que el AUC (área bajo la curva ROC) podrá tomar como valor máximo 1, la función de fitness devolverá $(1 - AUC)$.

Para poder calcular dicho área será necesario primero obtener la curva ROC, que la cual será representada mediante el método empírico explicado en la sección de las Curvas ROC. Se le tendrá que pasar por parámetros el conjunto de datos con los que realizar las pruebas, y un vector de pesos de la red a mejorar. Con ellos se podrá realizar la matriz de confusión necesaria para obtener la curva ROC de manera empírica.
Una vez obtenida la curva, será necesario obtener el área resultante bajo la misma, para ello se hará uso de la función trapezoidal, que cumple perfectamente con lo deseado.

5.1.2. Codificación del cromosoma

La optimización a realizar por parte del evolutivo exige que tenga una representación numérica de la red neuronal a mejorar. Es por ello que se necesita abstraer de la estructura de la red la configuración que determine su operacionalidad e incorporarla a un vector de pesos, que será el cromosoma del que partirá el genético para realizar su optimización, y que mantendrá su estructura, para así poder realizar evaluaciones con la función de fitness. El cromosoma tendrá tantas posiciones como pesos y umbrales tenga la red.

5.1.3. Validación cruzada

Ya que se quiere realizar un estudio lo más completo posible, se recurre a realizar validación cruzada con cada uno de los dominios a estudiar. La validación cruzada consiste en desglosar cada conjunto de datos en un número determinado de grupos o folds para que cuando se realice el entrenamiento, con un mismo conjunto de datos se puedan realizar todas las combinaciones de entrenamiento y test. Esto quiere decir, que si se decide hacer una validación cruzada de 5 folds, se sucederán 5 iteraciones, en las que el subconjunto de test irá cambiando, la primera iteración los conjuntos 1,2,3 y 4 se usarán para el entrenamiento de la red y el 5 para testear los resultados; en la siguiente iteración los conjuntos 1,2,3 y 5 serán los encargados del entrenamiento y el 4 del test. Así se sucederán las iteraciones hasta la última

donde los conjuntos 2,3,4 y 5 se utilizarán para el entrenamiento y el 1 para el test.

Para este trabajo se tiene que tener especial cuidado con la validación cruzada, ya que al realizar el estudio con dominios desbalanceados, si alguno de los folds por aleatoriedad se queda con un gran número de instancias de una clase poco entrenada, los resultados serán pésimos ya que el entrenamiento de la red de poco serviría, o incluso podría pasar que alguno llegase a quedarse sin instancias de la clase minoritaria. Para ello se realizará una distribución estratificada de las clases entre los diferentes folds mediante el uso de semillas distintas para cada clase, que indicarán en función de la clase de cada instancia, al fold al que pertenece, de esta manera se produce una división aleatoria entre el número de grupos de validación cruzada, pero sin perder el porcentaje de clases existente en el conjunto inicial para cada uno de los subconjuntos generados.

5.2. Herramientas de programación

Para el desarrollo del sistema se usará Matlab. Este software se ha elegido por su implementación eficiente de CMAES además de por su eficiencia en el uso y cálculo con matrices.

5.2.1. Codificación del sistema

El siguiente algoritmo ha sido desarrollado siguiendo las pautas anteriormente descritas y con objetivo de cumplir con la intención de esta investigación. Se usarán tres variables globales, globalNET, globalINPUTS, globalOUTPUTS, encargadas de transferir datos entre las distintas funciones que no se encuentren en el mismo espacio de trabajo.

Se comenzará con el algoritmo para calcular el área bajo la curva. Sus parámetros de entrada son las salidas que se esperan de la red, y las salidas otorgadas por la misma, ambas en un rango de 0 a 1. Para calcular el área es necesario calcular antes la propia curva ROC. Una vez hecho esto el área se obtiene fácilmente usando la función trapz (trapezoidal) de matlab.

```
1 function auc = AreaBajoCurvaRoc(salidas_esperadas, salidas)
2 [tpr, fpr, thresholdsTrainBoth] = roc(salidas_esperadas, salidas);
3 auc = trapz(fpr,tpr);
4 end
```

A continuación, se muestra la función de fitness. Será usada principalmente por CMAES, en la comparación de los individuos para seleccionar el más apto.
Por ellos, esta función implementa los pesos de la red indicados por parámetros a la red neuronal configurada al inicio de la ejecución. Realiza la simulación usando los datos indicados en la variable globalINPUTS. Los resultados de dicha simulación serán tratados para obtenerlos en un rango de 0 a 1 (el necesario para poder calcular la curva ROC), tras ello se llama a la función que calcula el área bajo la curva ROC indicada anteriormente, comunicando en sus parámetros de entrada, la salida obtenida de la simulación, y la que salida esperada para dicho paciente. Por último el valor de fitness que retornará será

la resta de la unidad menos el área bajo la curva, debido a que el objetivo es maximizar dicho área, y el objetivo de CMAES es minimizar el fitness. Realizando esta operación, cuanto mayor sea el área bajo la curva, menor será la respuesta de la función de fitness.

```
5  function f = fitnessFunction(pesos, lgscal, expon, expon2)
6  global globalNET globalINPUTS globalOUTPUTS;
7  net_fitness = globalNET;
8  net_fitness = setwb(net_fitness, pesos);
9  simulacion = sim(net_fitness,globalINPUTS');
10 outputs2 = (globalOUTPUTS+1)/2; simulacion2 = (simulacion+1)/2;
11 f = 1 - AreaBajoCurvaRoc(outputs2', simulacion2);
12 end
```

CMAES

El código del algoritmo evolutivo utilizado para este trabajo es de dominio público y puede encontrarse para su descarga en el siguiente enlace. `https://www.lri.fr/~hansen/cmaes.m` Este algoritmo evolutivo necesita un cromosoma inicial con el que empezar su ejecución, que en este caso será el vector de pesos de la red neuronal obtenido hasta el momento. Y cada vez que requiera realizar una evaluación entre dos individuos ejecutará la función de fitness anteriormente definida.

ExecuteFeedforwardNet Mediante esta función la red neuronal se configurará con los parámetros de entrada indicados, además de eso, en el código se indica el porcentaje de datos que se utilizará para entrenamiento, así como para test y validación. Para la función de entrenamiento se recurrirá a la por defecto de matlab (trainlm), se indican un máximo de 6000 ciclos y una tasa de aprendizaje de 0.125. Además se indica que la función de transferencia sea la tangente sigmoidal, de esta manera se asegura que la salida de la red esté comprendida entre los valores -1 y 1. Para mayor comodidad de la ejecución del programa se evita que se muestre el informe del estado del perceptrón multicapa. Por último se entrena la red usando la función train. Una vez entrenada se retorna la propia red.

```
13 function net = executeFeedforwardNet(trainMatrixInputs, trainMatrixOutputs, neuronas)
14 net = fitnet(neuronas);
15 net.divideParam.trainRatio = 0.8;
16 net.divideParam.testRatio = 0;
17 net.divideParam.valRatio = 0.2;
18 net.trainFcn = 'trainlm';
19 net.trainParam.epochs = 6000;
20 net.trainParam.lr = 0.125;
21 net.layers{2}.transferFcn = 'tansig';
22 net.trainParam.showWindow = false;
23 net = train(net,trainMatrixInputs,trainMatrixOutputs);
24 end
```

El orden del procedimiento exige que se ejecute primero la red neuronal con el conjunto de datos destinados al entrenamiento, tras obtener los pesos de la red entrenada, estos pasan como cromosoma de un individuo a algoritmo evolutivo, donde su objetivo será optimizar la exactitud de las pruebas maximizando el área bajo la curva ROC. Dado que para realizar una correcta evaluación de los resultados se ha decidido realizar validación cruzada, los datos para entrenamiento y test han de ser seleccionados en cada iteración de dicha validación.

Para aumentar la comodidad a la hora de consultar las pruebas, se decide almacenar en las variables

"weigths_values"los pesos de la red antes de ser optimizada por el evolutivo, en "fitness_values"los valores de fitness obtenidos por la red con los pesos indicados en la anterior variable para el conjunto de datos de entrenamiento, en "fitness_values_test"los valores de fitness obtenidos por la misma red con los mismos pesos antes indicados pero para el conjunto de datos de test. De la misma manera "weigths_values_gen", "fitness_gen_valuesz "fitness_gen_values_test", contienen la misma información que sus respectivos, pero obtenida una vez el algoritmo evolutivo ha optimizado los valores de los pesos de la red neuronal.
Como matiz final, para favorecer más si cabe la consulta de la información de cada ejecución del algoritmo se salvan las variables en un fichero .mat, cuyo nombre vendrá dado por 'SAVEDVARS_',nombre del dominio,'_',iteración de la validación cruzada,'_',semilla utilizada,'.mat'.

```
25
26  fitness_values = zeros(nFolds,1);
27  fitness_gen_values = zeros(nFolds,1);
28  fitness_values_test = zeros(nFolds,1);
29  fitness_gen_values_test = zeros(nFolds,1);
30  weigths_values = [];
31  weigths_values_gen = [];
32  for fold = 1:nFolds,
33  echo = ['Caso ',num2str(fold),':'];
34  disp(echo);
35  trainMatrix = getTrainMatrix(crossValidationMatrix, fold);
36  testMatrix = getTestMatrix(crossValidationMatrix, fold);
37  trainMatrix = randomizeMatrix(trainMatrix);
38  testMatrix = randomizeMatrix(testMatrix);
39
40
41  trainMatrixInputs = trainMatrix(:,1:end-1);
42  trainMatrixOutputs = trainMatrix(:,end:end);
43  trainMatrixOutputs = (trainMatrixOutputs - 1)*2 - 1;
44
45  net = executeFeedforwardNet(trainMatrixInputs', trainMatrixOutputs', neuronas);
46
47  testMatrixInputs = testMatrix(:,1:end-1);
48  testMatrixOutputs = testMatrix(:,end:end);
49  testMatrixOutputs = (testMatrixOutputs - 1)*2 - 1;
50
51  globalNET = net;
52  globalINPUTS = trainMatrixInputs;
53  globalOUTPUTS = trainMatrixOutputs;
54  weigths_values(:,fold) = getwb(net);
55  fitness_values(fold,1) = fitnessFunction(getwb(net));
56  fitness_values(fold,1)
57
58
59  globalNET = net;
60  globalINPUTS = testMatrixInputs;
61  globalOUTPUTS = testMatrixOutputs;
62  weigths_values(:,fold) = getwb(net);
63  fitness_values_test(fold,1) = fitnessFunction(getwb(net));
64  fitness_values_test(fold,1)
65
```

```
66
67
68  X = getwb(net);
69
70  globalINPUTS = trainMatrixInputs;
71  globalOUTPUTS = trainMatrixOutputs;
72  insigma = 1.5;
73  [XMIN,fmin,counteval,stopflag,out,bestever] =
74  cmaes('fitnessFunction', X, insigma);
75  bestever = bestever.x;
76  setwb(net,bestever);
77  weigths_values_gen(:,fold) = getwb(net);
78
79  globalNET = net;
80  globalINPUTS = trainMatrixInputs;
81  globalOUTPUTS = trainMatrixOutputs;
82  fitnessGenTest = fitnessFunction(bestever);
83
84
85  fitness_gen_values(fold,1) = fitnessGenTest;
86  fitness_gen_values(fold,1)
87
88
89
90  globalNET = net;
91  globalINPUTS = testMatrixInputs;
92  globalOUTPUTS = testMatrixOutputs;
93  fitnessGenTest = fitnessFunction(bestever);
94
95  fitness_gen_values_test(fold,1) = fitnessGenTest;
96  fitness_gen_values_test(fold,1)
97  fileNameToSave =
98  strcat('SAVEDVARS_',file_path,'_',int2str(fold),'_',int2str(seed),'.mat');
99  save(fileNameToSave);
100 end
```

Como se puede apreciar, para el fragmento de código anterior, son necesarias las siguientes funciones encargadas de seleccionar el conjunto de datos indicado para cada fold, por cada una de las iteraciones que requiera la validación cruzada. Seleccionando así el conjunto de entrenamiento o test (según la función que se llame) del fold concreto, teniendo en cuenta que estará indicado en la primera columna de dicha matriz. También se requiere de la función randomizeMatrix, encargada de alterar la ordenación de la matriz.

```
101 function trainMatrix = getTrainMatrix(matrix, foldToExclude)
102 trainMatrix = matrix(matrix(:, 1) ~= foldToExclude, 2 : end );
103 end
104
105 function testMatrix = getTestMatrix(matrix, fold)
106 testMatrix = matrix(matrix(:, 1) == fold, 2 : end);
107 end
108
109 function matrix = randomizeMatrix(matrix)
```

```
idx = randperm(size(matrix,1));
matrix = matrix(idx, :);
end
```

Para que la matriz de trabajo tenga la estructura deseada es necesario realizar un preprocesamiento de los datos. Los requisitos para que la matriz prometa una correcta compatibilidad con el resto del código son, que las filas indiquen en las primeras columnas los datos de entrada para la red por cada uno de los pacientes, y en la última columna se indique el resultado esperado para cada paciente, 0 o 1.
además la primera columna debe indicar a qué fold pertenece dicho paciente. Para una correcta validación cruzada, es imprescindible que el porcentaje de individuos con los diferentes estados, sea lo más equitativo posible por cada agrupación.
Mediante la siguiente porción de código se da solución a los requisitos descritos, obteniendo así, una matriz preparada para la validación cruzada y para una correcta ejecución del programa.

```
rng('default');
matrix = load(file_path);
matrixSize = size(matrix);
matrixRowsNumber = matrixSize(1,1);
matrixColumnsNumber = matrixSize(1,2);
matrix = sortrows(matrix, matrixColumnsNumber);
currentClass = matrix(1,matrixColumnsNumber);
foldVector = zeros(matrixRowsNumber,1);
for currentRow = 1:matrixRowsNumber,
if currentClass ~= matrix(currentRow,matrixColumnsNumber);
currentClass = matrix(currentRow,matrixColumnsNumber);
seed = seed + 1;
rng(seed);
end
fold = floor(rand() * nFolds) + 1;
foldVector(currentRow,1) = fold;
end

crossValidationMatrix = [foldVector matrix];
```

Capítulo 6

Validación experimental

6.1. Proceso

Gracias al diseño realizado, la ejecución de los experimentos será sencilla, ya que tan sólo habrá que ejecutar el código anteriormente indicado y seleccionar el dominio con el que se desea realizar la experimentación, así como diferentes parámetros que se indicarán en un próximo apartado. Tras la ejecución del sistema de optimización se obtendrán diferentes variables en estructura matricial con información sobre los pesos antes y después de entrenar la red, tras optimizarla con el genético, y el área bajo la curva ROC conseguido tanto para el conjunto de entrenamiento como para el conjunto de test y para cada uno de los grupos de la validación cruzada. Con estos resultados se puede proceder a su valoración.

6.2. Descripción de los dominios

Para realizar la experimentación, en este proyecto se han utilizado 5 dominios de clasificación diferentes, todos ellos desbalanceados en mayor o menor medida. A continuación se describen sus características.

OIL Este dominio consta de 50 variables de entrada y 1 variable de salida. Ésta última representa la clase y toma valores 1 (Clase 1) y 2 (Clase 2). Tiene 937 instancias con una proporción de clases 95.62 % de la clase 1 frente al 4.38 % de la clase 2. Se trata de un dominio altamente desbalanceado.

GERMAN.NUMER Este DOMINIO consta de 24 variables de entrada y 1 variable de salida. Ésta última representa la clase y toma valores 1 (Clase 1) y 2 (Clase 2). Tiene 1000 instancias con una proporción de clases 70 % de la clase 1 frente al 30 % de la clase 2.

ESTATE Este dominio consta de 12 variables de entrada y 1 variable de salida, que toma valores 1 y 2 para representar las 2 clases. Se dispone de 5322 instancias con una proporción de clases 88.05 % de la

clase 1 frente al 11.95 % de la clase 2.

DIABETES Este fichero consta de 8 variables de entrada y 1 como salida, que representa la clase, esta última está comprendida entre 1 y 2. Tiene 768 instancias con una proporción de clases 34.90 % de la clase 1 frente al 65.10 % de la clase 2. De entre todos los dominios, éste es el menos desbalanceado.

BLOOD Este dominio consta de 4 variables de entrada y 1 variable de salida, que representa la clase, y que toma valores 1 y 2. Tiene 748 instancias con una proporción de clases 76.20 % de la clase 1 frente al 23.80 % de la clase 2.

Los datos de todos estos dominios han sido normalizados en el intervalo $[0, 1]$ para una correcta integración con la red.

6.3. Parámetros utilizados

Para realizar la experimentación es necesario definir un conjunto de parámetros, los cuales se especifican a continuación:

Número de neuronas Indicando el número de neuronas que se desean en la capa oculta del perceptrón multicapa. Para la experimentación de este proyecto tomará el valor de 5 y 10.

Semilla El sistema desarrollado implica la aleatorización de diferentes variables, que afectarán a la repartición de instancias entre los distintos grupos destinados a validación cruzada, como para la inicialización de los pesos de la red. Para ello se define una semilla aleatoria inicial que será utilizada para todos los procesos aleatorios que conllevan la ejecución del sistema. En este proyecto se utilizaran tres valores diferentes para la semilla inicial, con el objetivo de estudiar la generalización del sistema. Las valores utilizados han sido 65885, 280919 y 398576.

Número de ciclos Número de ciclos máximos para realizar el aprendizaje del perceptrón multicapa. Este número se ha fijado a 6000. Debido a que las redes de matlab establecen un criterio de parada, éste número de ciclos se alcanzaría en los casos más adversos y no ha hecho falta alcanzarlo al haber cumplido en todas las experimentaciones el criterio de parada de matlab. Este criterio esta conformado de tres medidas que pueden realizar la parada del aprendizaje: Cuando el error llega a ser 0. Cuando el gradiente toma un valor inferior a 10^{-7}, a medida que el error cuadrático de la red va disminuyendo, el gradiente se hace cada vez menor. Si el número de validaciones sin disminuir el error llega a 6 seguidas.

Ratio de entrenamiento Porcentaje que indica la cantidad de datos que se usarán del conjunto pasado a la red para su entrenamiento. Con el 80 % de los datos se procederá a realizar el entrenamiento

Ratio de validación Este porcentaje indica la cantidad de instancias que se utilizarán como conjunto de validación, que determinará la parada del proceso de entrenamiento de la red en caso de suceder por el criterio de máximo número de validaciones sin disminuir el error. Para las validaciones se utilizará un 20 % del conjunto de datos destinado al entrenamiento de la red.

Razón de aprendizaje Indica la razón de aprendizaje para llevar a cabo el entrenamiento de la red. Para este trabajo se ha establecido 0.125 como factor de aprendizaje.

insigma Es un parámetro utilizado por CMAES que indica la desviación típica inicial, que suele estar en torno a 2. para este trabajo se le ha otorgado un valor de 1.5

6.4. Resultados obtenidos

A continuación se muestran los resultados obtenidos por el sistema desarrollado. Se han realizado pruebas experimentales para 5 y 10 neuronas, utilizando como población inicial para CMAES una red con pesos inicializados aleatoriamente o una red ya entrenada con el algoritmo de retropropagación. Para cada una de estas combinaciones se han realizado 3 ejecuciones variando la semilla inicial de la que partirán todos los cálculos aleatorios del sistema.

De manera organizada en diferentes subsecciones, se muestran los resultados obtenidos para cada uno de los dominios utilizados. En los cuales se muestran las medias para los resultados obtenidos en los diferentes grupos de la validación cruzada, así como la desviación típica.

6.4.1. Oil

A continuación se mostrarán los resultados obtenidos a partir de diferentes experimentaciones para el dominio Oil.

- **La tabla 6.1** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.2** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.3** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.4** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.5** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.6** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.7** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.8** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.

Población inicializada con Red Neuronal Entrenada, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.7356	0.7193	0.9985	0.8308
	Semilla 2	0.7983	0.7937	0.9978	0.9340
	Semilla 3	0.7549	0.7215	0.9977	0.8980
Desv. Típica	**Semilla 1**	0.3043	0.2588	0.0004	0.0676
	Semilla 2	0.2775	0.2538	0.0012	0.0464
	Semilla 3	0.3779	0.3666	0.0019	0.0644

DATOS 6.1: OIL, RED ENTRENADA 5 NEURONAS

		RNA	**EVOL**	**Diff**
Media Total	**Train**	0.7629	0.9980	0.2350
	Test	0.7448	0.8876	0.1428
Media Desv. Típica	**Train**	0.3199	0.0012	0.3187
	Test	0.2931	0.0595	0.2336

DATOS 6.2: MEDIA OIL, RED ENTRENADA 5 NEURONAS

Población inicial aleatoria, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.5253	0.6057	0.9986	0.7911
	Semilla 2	0.5304	0.4466	0.9993	0.8298
	Semilla 3	0.4245	0.3619	0.9985	0.8864
Desv. Típica	**Semilla 1**	0.0608	0.0644	0.0009	0.1248
	Semilla 2	0.0486	0.0557	0.0006	0.0967
	Semilla 3	0.1365	0.1982	0.0011	0.0357

DATOS 6.3: OIL, RED ALEATORIA 5 NEURONAS

		RNA	**EVOL**	**Diff**
Media Desv. Típica	**Train**	0.4934	0.9988	0.5054
	Test	0.4714	0.8358	0.3644
Media Desv. Típica	**Train**	0.0820	0.0009	0.0811
	Test	0.1061	0.0857	0.0204

DATOS 6.4: MEDIA OIL, RED ALEATORIA 5 NEURONAS

Población inicializada con Red Neuronal Entrenada, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.9493	0.9049	0.9993	0.8368
	Semilla 2	0.8065	0.8393	0.9995	0.8608
	Semilla 3	0.9740	0.8406	0.9987	0.8268
Desv. Típica	**Semilla 1**	0.0288	0.0748	0.0004	0.0979
	Semilla 2	0.2104	0.1139	0.0005	0.1006
	Semilla 3	0.0120	0.1358	0.0008	0.0815

DATOS 6.5: OIL, RED ENTRENADA 10 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.9099	0.9992	0.0892
	Test	0.8616	0.8415	-0.0201
Media Desv. Típica	**Train**	0.0837	0.0006	0.0832
	Test	0.1082	0.0933	0.0148

DATOS 6.6: MEDIA OIL, RED ENTRENADA 10 NEURONAS

Población inicial aleatoria, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.4752	0.4850	0.9992	0.8528
	Semilla 2	0.4988	0.5192	0.9994	0.7754
	Semilla 3	0.4679	0.4647	0.9994	0.8129
Desv. Típica	**Semilla 1**	0.1107	0.1787	0.0002	0.0560
	Semilla 2	0.0523	0.1132	0.0004	0.0803
	Semilla 3	0.1111	0.1312	0.0003	0.1304

DATOS 6.7: OIL, RED ALEATORIA 10 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.4806	0.9993	0.5187
	Test	0.4896	0.8137	0.3241
Media Desv. Típica	**Train**	0.0914	0.0003	0.0911
	Test	0.1410	0.0889	0.0521

DATOS 6.8: MEDIA OIL, RED ALEATORIA 10 NEURONAS

Para el dominio de OIL usando una población inicializada con una red neuronal de 5 neuronas en su capa oculta y entrenada, se puede ver que el área bajo la curva ROC calculada para la red neuronal tras su entrenamiento tiene valores en torno a **0.76** al evaluar el conjunto de entrenamiento y a **0.74** al evaluar el conjunto de test. Tras la optimización realizada por el algoritmo evolutivo a partir de la red ya entrenada (lo que significa que como mínimo obtendrá para los conjuntos de entrenamiento los mismos resultados) se obtiene un área bajo la curva de **0.99** en entrenamiento y de **0.88** en test. Por tanto, la mejora obtenida al haber realizado la optimización ha sido de **23 %** para el conjunto de entrenamiento y de **14 %** para el conjunto de test, consiguiendo así que el proceso de optimización evolutivo mejore los resultados. Además cabe añadir otro aspecto positivo, se pueden apreciar como la desviación típica tras la optimización por medio del evolutivo es muy baja, esto es debido a la poca variabilidad en los resultados del área obtenidos para cada uno de los folds.

Cuando usando la misma estructura de red (5 neuronas en la capa oculta), sólo se inicializan sus pesos aleatoriamente y se procede a la optimización con el evolutivo, los resultados de la red son irrelevantes, y la información que se puede sacar de ellos es verificar que se inicializó correctamente con valores aleatorios, consiguiendo así en evaluación del área bajo la curva ROC resultados pésimos. Inferiores a 0.5. El evolutivo sobre la red inicializada ha conseguido buenos valores, **0.99** para el conjunto de entrenamiento y **0.83** para el de test. Estos valores son buenos, pero en el anterior caso donde la red había sido previamente entrenada, se obtuvo por encima de un $5\,\%$ más de área bajo la curva que sin entrenarla.

Usando una población inicializada con red neuronal de 10 neuronas en su capa oculta y entrenada, se pueden apreciar muy buenos resultados al igual que con el experimento realizado con 5 neuronas anteriormente. Aunque en este caso, los resultados obtenidos por la red tras el entrenamiento evaluando los conjuntos de test, son mejores que tras la optimización haciendo uso del evolutivo. Esto puede ser debido a que haya ocurrido sobreaprendizaje.

Usando la misma estructura de red anterior (10 neuronas) pero con pesos inicializados aleatoriamente, se vuelve a apreciar los malos resultados para la red sin entrenar, antes de la optimización con el evolutivo. En este caso los resultados obtenidos pese a ser buenos, son inferiores a los conseguidos en el caso anterior.

6.4.2. german.numer

A continuación se mostrarán los resultados obtenidos a partir de diferentes experimentaciones para el dominio German.numer.

- **La tabla 6.9** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.10** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.11** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.12** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.13** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.14** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.15** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.16** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.

Población inicializada con Red Neuronal Entrenada, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.8235	0.7563	0.9005	0.7460
	Semilla 2	0.8217	0.7378	0.8942	0.7365
	Semilla 3	0.8005	0.7509	0.8961	0.7523
Desv. Típica	**Semilla 1**	0.0240	0.0212	0.0083	0.0323
	Semilla 2	0.0192	0.0452	0.0097	0.0360
	Semilla 3	0.0258	0.0269	0.0045	0.0339

DATOS 6.9: GERMAN.NUMER, RED ENTRENADA 5 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.8152	0.8969	0.0817
	Test	0.7483	0.7449	-0.0034
Media Desv. Típica	**Train**	0.0230	0.0075	0.0155
	Test	0.0311	0.0341	-0.0030

DATOS 6.10: MEDIA GERMAN.NUMER, RED ENTRENADA 5 NEURONAS

Población inicial aleatoria, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.4765	0.4931	0.9000	0.7301
	Semilla 2	0.4750	0.5194	0.8894	0.7551
	Semilla 3	0.5307	0.5420	0.8986	0.7554
Desv. Típica	**Semilla 1**	0.1050	0.1340	0.0032	0.0232
	Semilla 2	0.1128	0.0995	0.0065	0.0162
	Semilla 3	0.0707	0.0776	0.0085	0.0528

DATOS 6.11: GERMAN.NUMER, RED ALEATORIA 5 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.4941	0.8960	0.4019
	Test	0.5182	0.7469	0.2287
Media Desv. Típica	**Train**	0.0962	0.0061	0.0901
	Test	0.1037	0.0307	0.0730

DATOS 6.12: MEDIA GERMAN.NUMER, RED ALEATORIA 5 NEURONAS

Población inicializada con Red Neuronal Entrenada, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.8163	0.7505	0.9419	0.7150
	Semilla 2	0.7972	0.7098	0.9401	0.7461
	Semilla 3	0.8100	0.7688	0.9343	0.7325
Desv. Típica	**Semilla 1**	0.0289	0.0416	0.0070	0.0423
	Semilla 2	0.0728	0.0926	0.0039	0.0282
	Semilla 3	0.0150	0.0470	0.0075	0.0384

DATOS 6.13: GERMAN.NUMER, RED ENTRENADA 10 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.8078	0.9388	0.1309
	Test	0.7430	0.7312	-0.0118
Media Desv. Típica	**Train**	0.0389	0.0061	0.0328
	Test	0.0604	0.0363	0.0241

DATOS 6.14: MEDIA GERMAN.NUMER, RED ENTRENADA 10 NEURONAS

Población inicial aleatoria, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.4742	0.4574	0.9444	0.7313
	Semilla 2	0.4877	0.4669	0.9384	0.7324
	Semilla 3	0.4724	0.4911	0.9321	0.7362
Desv. Típica	**Semilla 1**	0.0491	0.0654	0.0052	0.0167
	Semilla 2	0.0670	0.0548	0.0031	0.0282
	Semilla 3	0.0993	0.1168	0.0120	0.0466

DATOS 6.15: GERMAN.NUMER, RED ALEATORIA 10 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.4781	0.9383	0.4602
	Test	0.4718	0.7333	0.2615
Media Desv. Típica	**Train**	0.0718	0.0068	0.0650
	Test	0.0790	0.0305	0.0485

DATOS 6.16: MEDIA GERMAN.NUMER, RED ALEATORIA 10 NEURONAS

Para el dominio de GERMAN.NUMER usando una población inicializada con una red neuronal de 5 neuronas en su capa oculta y entrenada, se puede ver que el área bajo la curva ROC calculada para la red neuronal tras su entrenamiento tiene valores en torno a **0.81** al evaluar el conjunto de entrenamiento y a **0.74** al evaluar el conjunto de test. Tras la optimización realizada por el algoritmo evolutivo a partir de la red ya entrenada (lo que significa que como mínimo obtendrá para los conjuntos de entrenamiento los mismos resultados) se obtiene un área bajo la curva de **0.89** en entrenamiento y de **0.74** en test. Por tanto, la mejora obtenida al haber realizado la optimización ya que la evaluación del conjunto de test apenas ha variado, es útil tan solo para el conjunto de entrenamiento.

Al igual que se comenta en el apartado anterior, cuando usando la misma estructura de red (5 neuronas en la capa oculta), sólo se inicializan sus pesos aleatoriamente y se procede a la optimización con el evolutivo, los resultados de la red son irrelevantes, y la información que se puede sacar de ellos es verificar que se inicializó correctamente con valores aleatorios, consiguiendo así en evaluación del área bajo la curva ROC resultados pésimos. Inferiores a 0.5. El evolutivo sobre la red inicializada ha conseguido buenos valores, **0.89** para el conjunto de entrenamiento y **0.74** para el de test. Estos valores son buenos, pero no se aprecia diferencia alguna con los obtenidos tras la optimización en el anterior caso.

Usando una población inicializada con red neuronal de 10 neuronas en su capa oculta y entrenada, se pueden apreciar muy buenos resultados al igual que con el experimento realizado con 5 neuronas anteriormente. Aunque en este caso, los resultados obtenidos por la red tras el aprendizaje evaluando los conjuntos de entrenamiento, son mejores que tras la optimización haciendo uso del genético. Pero los obtenidos para el conjunto de test, aprecian un leve empeoramiento, de donde se puede deducir, que la optimización se ha centrado demasiado en el conjunto de entrenamiento, ha vuelto a realizar sobreaprendizaje con 10 neuronas.

Usando la misma estructura de red anterior (10 neuronas) pero con pesos inicializados aleatoriamente, se vuelve a apreciar los malos resultados para la red sin entrenar, antes de la optimización con el evolutivo. En este caso los resultados obtenidos pese a ser buenos, son prácticamente idénticos a los obtenidos en el caso anterior.

6.4.3. estate

A continuación se mostrarán los resultados obtenidos a partir de diferentes experimentaciones para el dominio Estate.

- **La tabla 6.17** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.18** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.19** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.20** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.21** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.22** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.23** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.24** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.

Población inicializada con Red Neuronal Entrenada, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.6343	0.6319	0.6640	0.6274
	Semilla 2	0.6368	0.6241	0.6660	0.6384
	Semilla 3	0.6362	0.6220	0.6716	0.6394
Desv. Típica	**Semilla 1**	0.0118	0.0239	0.0169	0.0334
	Semilla 2	0.0134	0.0080	0.0038	0.0106
	Semilla 3	0.0095	0.0211	0.0097	0.0187

DATOS 6.17: ESTATE, RED ENTRENADA 5 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.6357	0.6672	0.0314
	Test	0.6260	0.6350	0.0090
Media Desv. Típica	**Train**	0.0116	0.0101	0.0014
	Test	0.0177	0.0209	-0.0032

DATOS 6.18: MEDIA ESTATE, RED ENTRENADA 5 NEURONAS

Población inicial aleatoria, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.4639	0.4846	0.6678	0.6346
	Semilla 2	0.4583	0.4537	0.6645	0.6353
	Semilla 3	0.5107	0.5320	0.6650	0.6302
Desv. Típica	**Semilla 1**	0.0178	0.0243	0.0051	0.0274
	Semilla 2	0.0311	0.0255	0.0044	0.0225
	Semilla 3	0.0228	0.0397	0.0092	0.0217

DATOS 6.19: ESTATE, RED ALEATORIA 5 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.4776	0.6658	0.1881
	Test	0.4901	0.6334	0.1433
Media Desv. Típica	**Train**	0.0239	0.0062	0.0177
	Test	0.0298	0.0239	0.0060

DATOS 6.20: MEDIA ESTATE, RED ALEATORIA 5 NEURONAS

Población inicializada con Red Neuronal Entrenada, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.6403	0.6338	0.6769	0.6335
	Semilla 2	0.6443	0.6355	0.6762	0.6370
	Semilla 3	0.6398	0.6253	0.6836	0.6424
Desv. Típica	**Semilla 1**	0.0099	0.0140	0.0054	0.0151
	Semilla 2	0.0161	0.0095	0.0054	0.0103
	Semilla 3	0.0127	0.0218	0.0098	0.0301

DATOS 6.21: ESTATE, RED ENTRENADA 10 NEURONAS

		RNA	**EVOL**	**Diff**
Media Total	**Train**	0.6415	0.6789	0.0374
	Test	0.6315	0.6376	0.0061
Media Desv. Típica	**Train**	0.0129	0.0069	0.0060
	Test	0.0151	0.0185	-0.0034

DATOS 6.22: MEDIA ESTATE, RED ENTRENADA 10 NEURONAS

Población inicial aleatoria, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.4815	0.4883	0.6741	0.6366
	Semilla 2	0.5417	0.5279	0.6782	0.6456
	Semilla 3	0.5279	0.5240	0.6747	0.6309
Desv. Típica	**Semilla 1**	0.0330	0.0370	0.0133	0.0272
	Semilla 2	0.0397	0.0440	0.0039	0.0206
	Semilla 3	0.0633	0.0740	0.0055	0.0133

DATOS 6.23: ESTATE, RED ALEATORIA 10 NEURONAS

		RNA	**EVOL**	**Diff**
Media Total	**Train**	0.5170	0.6757	0.1586
	Test	0.5134	0.6377	0.1243
Media Desv. Típica	**Train**	0.0453	0.0076	0.0378
	Test	0.0517	0.0204	0.0313

DATOS 6.24: MEDIA ESTATE, RED ALEATORIA 10 NEURONAS

Para el dominio de ESTATE usando una población inicializada con una red neuronal de 5 neuronas en su capa oculta y entrenada, se puede ver que el área bajo la curva ROC calculada para la red neuronal tras su entrenamiento tiene valores en torno a **0.63** al evaluar el conjunto de entrenamiento y a **0.62** al evaluar el conjunto de test. Tras la optimización realizada por el algoritmo evolutivo a partir de la red ya entrenada (lo que significa que como mínimo obtendrá para los conjuntos de entrenamiento los mismos resultados) se obtiene un área bajo la curva de **0.66** en entrenamiento y de **0.63** en test. Por tanto la optimización ha provocado levemente una mejora tanto para el conjunto de entrenamiento como para el de test. La mejora obtenida al haber realizado la optimización ha sido de **3 %** para el conjunto de entrenamiento y de **0.9 %** para el de test.

Al igual que se comenta en el apartado anterior, cuando usando la misma estructura de red (5 neuronas en la capa oculta), sólo se inicializan sus pesos aleatoriamente y se procede a la optimización con el evolutivo, los resultados de la red son irrelevantes, y la información que se puede sacar de ellos es verificar que se inicializó correctamente con valores aleatorios, consiguiendo así en evaluación del área bajo la curva ROC resultados pésimos. Inferiores a 0.5. El evolutivo sobre la red inicializada ha conseguido unos valores altamente próximos a los obtenidos en el atnerior caso, **0.66** para el conjunto de entrenamiento y **0.63** para el de test. Estos valores son buenos, pero tan solo se aprecia una casi insignificante diferencia en el tercer decimal con respecto al anterior caso.

Usando una población inicializada con red neuronal de 10 neuronas en su capa oculta y entrenada, se pueden apreciar resultados aceptables, y en este caso, levemente mejores a los anteriores al haber obtenido para el conjunto de entrenamiento un área bajo la curva roc de **0.64** para el de test de **0.63**. Tras la optimización se logra **0.68** para el de entrenamiento y **0.64** para el de test, Lo que indica que para este dominio funciona mejor una red con 10 neuronas. La mejora obtenida al haber realizado la optimización ha sido de **3 %** para el conjunto de entrenamiento y de **0.6 %** para el de test.

Usando la misma estructura de red anterior (10 neuronas) pero con pesos inicializados aleatoriamente, se aprecian resultados cercanos a los obtenidos en el anterior caso, pero con algo más de desviación típica entre los resultados obtenidos por cada fold.

Los datos obtenidos dan a entender que se trata de un dominio de dificil clasificación, ya que apenas varía el área bajo al curva roc. Esto puede ser debido a su gran cantidad de instancias (5322) y a estar altamente desbalanceado, ya que tan solo el 11.95 % de las instancias pertenencen a la clase menos repersentativa.

6.4.4. Diabetes

A continuación se mostrarán los resultados obtenidos a partir de diferentes experimentaciones para el dominio Diabetes.

- **La tabla 6.25** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.26** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.27** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.28** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.29** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.30** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.31** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.32** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.

Población inicializada con Red Neuronal Entrenada, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.8554	0.8178	0.8775	0.8080
	Semilla 2	0.8221	0.7957	0.8835	0.8202
	Semilla 3	0.8622	0.8254	0.8794	0.8149
Desv. Típica	**Semilla 1**	0.0152	0.0446	0.0099	0.0360
	Semilla 2	0.0590	0.0906	0.0109	0.0315
	Semilla 3	0.0148	0.0263	0.0079	0.0306

DATOS 6.25: DIABETES, RED ENTRENADA 5 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.8465	0.8801	0.0335
	Test	0.8129	0.8143	0.0014
Media Desv. Típica	**Train**	0.0297	0.0096	0.0201
	Test	0.0538	0.0327	0.0211

DATOS 6.26: MEDIA DIABETES, RED ENTRENADA 5 NEURONAS

Población inicial aleatoria, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.4742	0.4411	0.8762	0.8260
	Semilla 2	0.4452	0.4465	0.8798	0.8094
	Semilla 3	0.4481	0.4588	0.8732	0.8207
Desv. Típica	**Semilla 1**	0.1493	0.1473	0.0097	0.0363
	Semilla 2	0.1650	0.1580	0.0102	0.0292
	Semilla 3	0.0340	0.0566	0.0050	0.0340

DATOS 6.27: DIABETES, RED ALEATORIA 5 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.4558	0.8764	0.4206
	Test	0.4488	0.8187	0.3699
Media Desv. Típica	**Train**	0.1161	0.0083	0.1078
	Test	0.1206	0.0332	0.0875

DATOS 6.28: MEDIA DIABETES, RED ALEATORIA 5 NEURONAS

Población inicializada con Red Neuronal Entrenada, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.8645	0.8158	0.9030	0.8061
	Semilla 2	0.8522	0.8155	0.9089	0.7968
	Semilla 3	0.7657	0.7445	0.9069	0.7985
Desv. Típica	**Semilla 1**	0.0314	0.0361	0.0130	0.0296
	Semilla 2	0.0127	0.0436	0.0104	0.0254
	Semilla 3	0.1828	0.1736	0.0102	0.0666

DATOS 6.29: DIABETES, RED ENTRENADA 10 NEURONAS

		RNA	**EVOL**	**Diff**
Media Total	**Train**	0.8275	0.9063	0.0788
	Test	0.7919	0.8005	0.0085
Media Desv. Típica	**Train**	0.0756	0.0112	0.0644
	Test	0.0844	0.0405	0.0439

DATOS 6.30: MEDIA DIABETES, RED ENTRENADA 10 NEURONAS

Población inicial aleatoria, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.3946	0.3905	0.9047	0.8156
	Semilla 2	0.4639	0.4451	0.9009	0.8161
	Semilla 3	0.4345	0.4201	0.9016	0.7996
Desv. Típica	**Semilla 1**	0.1218	0.1139	0.0069	0.0230
	Semilla 2	0.1553	0.1737	0.0048	0.0158
	Semilla 3	0.1455	0.1347	0.0069	0.0223

DATOS 6.31: DIABETES, RED ALEATORIA 10 NEURONAS

		RNA	**EVOL**	**Diff**
Media Total	**Train**	0.4310	0.9024	0.4714
	Test	0.4186	0.8104	0.3919
Media Desv. Típica	**Train**	0.1409	0.0062	0.1347
	Test	0.1408	0.0204	0.1204

DATOS 6.32: MEDIA DIABETES, RED ALEATORIA 10 NEURONAS

Para el dominio de DIABETES usando una población inicializada con una red neuronal de 5 neuronas en su capa oculta y entrenada, se puede ver que el área bajo la curva ROC calculada para la red neuronal tras su entrenamiento tiene valores en torno a **0.84** al evaluar el conjunto de entrenamiento y a **0.81** al evaluar el conjunto de test. Tras la optimización realizada por el algoritmo evolutivo a partir de la red ya entrenada (lo que significa que como mínimo obtendrá para los conjuntos de entrenamiento los mismos resultados) se obtiene un área bajo la curva de **0.88** en entrenamiento y de **0.81** en test. Por tanto, no hay casi mejora en test, aunque se logra una disminución en la variabilidad de resultados para distintos folds. La mejora obtenida al haber realizado la optimización ha sido de **3 %** para el conjunto de entrenamiento y de **0.1 %** para el de test.

Al igual que se comenta en el apartado anterior, cuando usando la misma estructura de red (5 neuronas en la capa oculta), sólo se inicializan sus pesos aleatoriamente y se procede a la optimización con el evolutivo, los resultados de la red son irrelevantes, y la información que se puede sacar de ellos es verificar que se inicializó correctamente con valores aleatorios, consiguiendo así en evaluación del área bajo la curva ROC resultados pésimos. Inferiores a 0.5. El evolutivo sobre la red inicializada ha conseguido buenos valores, **0.87** para el conjunto de entrenamiento y **0.81** para el de test. Estos valores son buenos, pero en el caso del conjunto de entrenamiento el área bajo la curva roc resultante es inferior a la del proceso anterior.

Usando una población inicializada con red neuronal de 10 neuronas en su capa oculta y entrenada, se consiguen resultados algo variables, ya que el conjunto de entrenamiento antes de optimizar la red disminuye en un $2\,\%$ respecto al obtenido en el anterior caso, al igual que el conjunto de test. Tras haber realizado la optimización de la red con el genético, para el conjunto de entrenamiento es capaz de obtener una curva de **0.90**, superior en $2\,\%$ al anterior caso, pero el conjunto de test, obtiene un resultado de 0.80. Por lo que sólo ha experimentado mejora el conjunto de entrenamiento tras la optmización. La mejora obtenida al haber realizado la optimización ha sido de **7 %** para el conjunto de entrenamiento y de **0.85 %** para el de test.

Usando la misma estructura de red anterior (10 neuronas) pero con pesos inicializados aleatoriamente, se obtiene un área bajo la curva roc similar a la obtenida en el anterior caso, pero el conjunto de test consigue llegar a obtener 0.81. Lo que da a entender que en el anterior caso el número de neuronas determinado hace mas proclive a la red a realizar sobreaprendizaje.

6.4.5. Blood

A continuación se mostrarán los resultados obtenidos a partir de diferentes experimentaciones para el dominio Blood.

- **La tabla 6.33** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.34** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.35** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 5 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.36** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.37** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red entrenada con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.38** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.
- **La tabla 6.39** muestra la media y desviación típica del área bajo la curva obtenida para cada una de las semillas con las que se desarrolla la experimentación usando una red inicializada aleatoriamente con 10 neuronas y después optimizada. Estos resultados se muestran para los conjuntos de entrenamiento y de test tanto de la red antes de optimizar como después.
- **La tabla 6.40** muestra la media total y desv típica entre semillas del área bajo la curva obtenida para el proceso anterior, tanto para la red antes de ser optimizada por el evolutivo, como después.

Población inicializada con Red Neuronal Entrenada, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.7667	0.7548	0.7777	0.7373
	Semilla 2	0.6799	0.6614	0.7778	0.7556
	Semilla 3	0.6105	0.5886	0.7784	0.7397
Desv. Típica	**Semilla 1**	0.0237	0.0810	0.0210	0.0910
	Semilla 2	0.1767	0.2123	0.0131	0.0690
	Semilla 3	0.3057	0.2952	0.0125	0.0477

DATOS 6.33: BLOOD, RED ENTRENADA 5 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.6857	0.7780	0.0923
	Test	0.6682	0.7442	0.0760
Media Desv. Típica	**Train**	0.1687	0.0155	0.1532
	Test	0.1962	0.0692	0.1269

DATOS 6.34: MEDIA BLOOD, RED ENTRENADA 5 NEURONAS

Población inicial aleatoria, 5 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.4667	0.4340	0.7798	0.7571
	Semilla 2	0.3685	0.3545	0.7761	0.7364
	Semilla 3	0.5373	0.5568	0.7851	0.7288
Desv. Típica	**Semilla 1**	0.0993	0.0943	0.0097	0.0232
	Semilla 2	0.0868	0.0610	0.0079	0.0326
	Semilla 3	0.1152	0.0916	0.0138	0.0248

DATOS 6.35: BLOOD, RED ALEATORIA 5 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.4575	0.7803	0.3228
	Test	0.4484	0.7408	0.2923
Media Desv. Típica	**Train**	0.1004	0.0105	0.0900
	Test	0.0823	0.0269	0.0554

DATOS 6.36: MEDIA BLOOD, RED ALEATORIA 5 NEURONAS

Población inicializada con Red Neuronal Entrenada, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.7713	0.7490	0.7935	0.7167
	Semilla 2	0.7690	0.7518	0.7968	0.7260
	Semilla 3	0.7700	0.7484	0.7929	0.7293
Desv. Típica	**Semilla 1**	0.0095	0.0371	0.0097	0.0412
	Semilla 2	0.0182	0.0643	0.0149	0.0632
	Semilla 3	0.0214	0.0541	0.0176	0.0623

DATOS 6.37: BLOOD, RED ENTRENADA 10 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.7701	0.7944	0.0243
	Test	0.7497	0.7240	-0.0257
Media Desv. Típica	**Train**	0.0164	0.0141	0.0023
	Test	0.0518	0.0556	-0.0037

DATOS 6.38: MEDIA BLOOD, RED ENTRENADA 10 NEURONAS

Población inicial aleatoria, 10 neuronas

		RNA		EVOL	
		Train	**Test**	**Train**	**Test**
Media	**Semilla 1**	0.4203	0.4369	0.7899	0.7461
	Semilla 2	0.4959	0.4862	0.7927	0.7188
	Semilla 3	0.4044	0.3952	0.7972	0.7398
Desv. Típica	**Semilla 1**	0.0768	0.1062	0.0091	0.0337
	Semilla 2	0.0822	0.0422	0.0077	0.0198
	Semilla 3	0.0510	0.0690	0.0094	0.0176

DATOS 6.39: BLOOD, RED ALEATORIA 10 NEURONAS

		RNA	EVOL	Diff
Media Total	**Train**	0.4402	0.7933	0.3531
	Test	0.4394	0.7349	0.2955
Media Desv. Típica	**Train**	0.0700	0.0087	0.0613
	Test	0.0725	0.0237	0.0488

DATOS 6.40: MEDIA BLOOD, RED ALEATORIA 10 NEURONAS

Para el dominio de BLOOD usando una población inicializada con una red neuronal de 5 neuronas en su capa oculta y entrenada, se puede ver que el área bajo la curva ROC calculada para la red neuronal tras su entrenamiento tiene valores en torno a **0.68** al evaluar el conjunto de entrenamiento y a **0.66** al evaluar el conjunto de test. Tras la optimización realizada por el algoritmo evolutivo a partir de la red ya entrenada (lo que significa que como mínimo obtendrá para los conjuntos de entrenamiento los mismos resultados) se obtiene un área bajo la curva de **0.78** en entrenamiento y de **0.74** en test. Por tanto, la mejora obtenida al haber realizado la optimización ha sido de **9 %** para el conjunto de entrenamiento y de **7 %** para el conjunto de test, consiguiendo así que el proceso de optimización evolutivo mejore los resultados. Además cabe añadir otro aspecto positivo, se pueden apreciar como la desviación típica tras la optimización por medio del genético es muy baja, esto es debido a la poca variabilidad en los resultados del área obtenidos para cada uno de los folds.

Al igual que se comenta en el apartado anterior, cuando usando la misma estructura de red (5 neuronas en la capa oculta), sólo se inicializan sus pesos aleatoriamente y se procede a la optimización con el evolutivo, los resultados de la red son irrelevantes, y la información que se puede sacar de ellos es verificar que se inicializó correctamente con valores aleatorios, consiguiendo así en evaluación del área bajo la curva ROC resultados pésimos. Inferiores a 0.5. El evolutivo sobre la red inicializada ha conseguido buenos valores, **0.78** para el conjunto de entrenamiento y **0.74** para el de test. Estos valores son buenos, y en comparativa con el caso anterior el conjunto de entrenamiento los supera levemente, y el de test, no los alcanza por muy pocos decimales.

Usando una población inicializada con red neuronal de 10 neuronas en su capa oculta y entrenada, se aprecia como los resultados mejoran para los conjuntos de entrenamiento tanto antes como después del evolutivo, además el conjunto de test antes de la optimización mejora en comparación con su respectivo para 5 neuronas, pero después de dicha optimización, el conjunto de test obtiene peores resultados. Para este dominio, se aprecia que con 10 neuronas se produce sobreaprendizaje.

Usando la misma estructura de red anterior (10 neuronas) pero con pesos inicializados aleatoriamente, se obtienen resultados similares a los obtenidos en el caso anterior, llegando a las mismas conclusiones.

Capítulo 7

Conclusiones y futuros trabajos

El objetivo de este proyecto era realizar una investigación para evaluar si las estrategias evolutivas son útiles como optimizadoras de la curva ROC para problemas desbalanceados, haciendo uso de la estructura de una red neuronal ya sea entrenada o inicializada con pesos aleatorios.

A continuación se mostrarán unas tablas resumen para los casos experimentados, de las que además de haber hecho su correspondiente evaluación para cada dominio, se podrán sacar conclusiones generalizadas.

	RED ENTRENADA		RED ALEATORIA	
	RNA	EVOL	RNA	EVOL
OIL	0.7448	0.8876	0.4714	0.8358
GERMAN.NUMER	0.7483	0.7449	0.5182	0.7469
ESTATE	0.6260	0.6350	0.4901	0.6334
DIABETES	0.8129	0.8143	0.4488	0.8187
BLOOD	0.6682	0.7442	0.4484	0.7408

DATOS 7.1: RESUMEN 5 NEURONAS

	RED ENTRENADA		RED ALEATORIA	
	RNA	EVOL	RNA	EVOL
OIL	0.8616	0.8415	0.4896	0.8137
GERMAN.NUMER	0.7430	0.7312	0.4718	0.7333
ESTATE	0.6315	0.6376	0.5134	0.6377
DIABETES	0.7919	0.8005	0.4186	0.8104
BLOOD	0.7497	0.7240	0.4394	0.7349

DATOS 7.2: RESUMEN 10 NEURONAS

Mediante este trabajo se ha probado como en dominios desbalanceados la optimización de la curva ROC para mejorar la clasificación mediante evolutivos es funcional. Queda demostrado en las tabla anteriores donde se puede apreciar que en la mayoría de los casos se realiza una mejora con respecto a los valores iniciales. En los casos que no lo hace, se puede observar al analizar en más detalle las tablas del

dominio concreto, cómo la desviación típica disminuye, lo que indica un intento de optimización frenado por un posible sobreentrenamiento.

Pensando en trabajos futuros, cabría considerar mejorar la estructura de la red, ya que en algunos de los experimentos se puede apreciar un notable sobreaprendizaje de la red.

Capítulo 8

Presupuesto

A continuación se procede al desglose del presupuesto ligado al proyecto desarrollado.

8.1. Cálculos de costes

8.1.1. Descripción del proyecto

Autor: Juan Carlos Pazos Mandiá

Departamento: Informática

Título: Optimización de curvas ROC para perceptrones multicapa mediante técnicas evolutivas

Duración: El proyecto se comenzó el 3 de Noviembre de 2014 y tiene como fecha de conclusión el 8 de Julio de 2015, al realizar su defensa. En total para el proyecto se han invertido 365 horas.

8.1.2. Costes de personal

Juan Carlos Pazos será el encargado de realizar todo el proyecto, a continuación se detall el desglose de las fases y su valoración económica.

Fase	Coste / hora	Total horas	Coste total
Análisis	30.00 €	40	1200.00 €
Diseño	40.00 €	60	2400.00 €
Codificación	45.00 €	125	5625.00 €
Pruebas	35.00 €	55	1925.00 €
Experimentación	30.00 €	40	1200.00 €
Documentación	40.00 €	45	1800.00 €
TOTAL		365	14150.00 €

DATOS 8.1: COSTE PERSONAL

8.1.3. Costes de equipos

Para la realización del trabajo ha sido necesaria la utilización de dos equipos, uno para desarrollo y otro para realizar la experimentación. El primero es un ordenador de las aulas de la Universidad Carlos III, el segundo es un equipo más potente, donde poder realizar de manera más eficaz las simulaciones.

Tipo de equipo	Unidades	Coste
Desarrollo	1	0€
Experimentación	1	835€
Total	2	835€

DATOS 8.2: COSTE DE EQUIPOS

8.1.4. Costes de software

El coste del Software del equipo de desarrollo no se tendrá en cuenta dado que está cedido por la Universidad Carlos III. Tanto el sistema operativo como el software de Matlab. El equipo de experimentación gracias al programa de colaboración de la universidad con MSDNAA posée como sistema Operativo Windows 8. Para el proceso de experimentación se ha usado el Runtime gratuito de Matlab. Para la documentación se ha utilizado TeXstudio, toda ella está escrita en latex, exportable al estandarizado formato de PDF.

Software	Unidades	Coste
S.O Equipo desarrollo	1	0€
S.O Equipo experimentación	1	0€
Matlab	1	0€
Matlab Runtime	1	0€
TeXstudio	1	0€
Total		0€

DATOS 8.3: COSTE DE SOFTWARE

8.1.5. Costes de material fungible

En la siguiente tabla se pueden apreciar los costes ocasionados por material fungible, como material de escritura, tinta de impresión, folios, y otros gastos (cafeína principalmente).

Han sido excluidos de esta estimación, los costes asumidos como la luz o el agua.

Concepto	Coste
Folios	7.6€
Material escritura	2.20€
Otros gastos	30€
Total	39.80€

DATOS 8.4: COSTE DE MATERIAL FUNGIBLE

8.2. Presupuesto

En la siguiente tabla se encuentra el compéndio de todos los costes anteriormente desglosados.

Concepto	**Coste total**
Coste personal	14150.00 €
Coste de equipo	835 €
Coste de software	0 €
Coste de material fungible	39.80 €
Total	15024.80 €
Total (incl. I.V.A.)	**18180.00 €**

DATOS 8.5: PRESUPUESTO TOTAL

El coste total del proyecto asciende a **DIECIOCHO MIL CIENTO OCHENTA EUROS** (18180.00 €)

Bibliografía

[1] Ricardo Aler, Inés M. Galván, José M. Valls. *Applying evolution strategies to preprocessing EEG signals for brain–computer interfaces.* 2012.

[2] Inés M. Galván León, Pedro Isasi Viñuela. *Redes de neuronas artificiales. Un enfoque práctico*. 2004.

[3] Laurence Fausett. *Fundamentals of Neural Networks*. 1994.

[4] Gray R, Begg CB, Greenes RA. *Construction of receiver operating characteristic curves when disease verification is subject to selection bias*. 1984.

[5] R.J. Hand, D.J., Till. *A simple generalization of the area under the ROC curve to multiple class classification problems.* 2001.

[6] Richard M. Everson, Jonathan E. Fieldsend. *Multi-class ROC analysis from a multi-objective optimisation perspective*. 2006.